QRD-BASED SQUARE ROOT FREE AND DIVISION FREE ALGORITHMS AND ARCHITECTURES

K.J.R. Liu E. Frantzeskakis
Electrical Engr. Dept.
Institute of Systems Research
University of Maryland
College Park, Maryland

Abstract - We introduce a family of square root free and division free rotation based algorithms. Our approach suggests a new perspective of the QR decomposition (QRD) algorithms and leads to a considerable reduction of the circuitry complexity and time delay in the associated architectures. The optimal residual and the optimal weight extraction for the recursive least squares (RLS) problem are considered in this paper. The systolic structures that are described are very promising, since they involve less computational complexity from the structures known to date and they make the VLSI implementation more tractable.

1 INTRODUCTION

The least squares (LS) minimization problem constitutes the core of many real-time signal processing problems, such as adaptive filtering, system identification and beamforming [4]. The most common version of the LS problem for adaptive signal processing is as follows: Solve the minimization problem

$$w(n) = \arg\min_{w(n)} \| \mathcal{B}(n)(X(n)w(n) - y(n)) \|^2, \tag{1}$$

where $X(n)$ is a matrix of size $n \times p$, $w(n)$ is a vector of length p, $y(n)$ is a vector of length n and $\mathcal{B}(n) = \text{diag}\{\beta^{n-1}, \beta^{n-2}, \cdots, 1\}, 0 < \beta < 1$, that is, β is a forgetting factor. There are two different pieces of information that may be required as the outcome result of this minimization [4]: The optimizing weight vector $w(n)$ and/or the optimal residual at the time instant n: $e_{RLS}(t_n) = X(t_n)w(n) - y(t_n)$, where $X(t_n)$ is the last row of the matrix $X(n)$ and $y(t_n)$ is the last element of the vector $y(n)$.
Efficient implementations of the recursive least squares (RLS) algorithms based on the QR decomposition (QRD) and the associated architectures were first introduced by McWhirter [6]. A comprehensive description of the algorithms and the architectural implementations is given in [4, chap.14]. The optimal weight vector $w(n)$ in (1) and the optimal residual are given by the

expressions

$$w^T(n) = u^T(n)\left(R^{-H}(n)\right)^* \tag{2}$$

and

$$e_{RLS}(t_n) = -\left(\prod_{k=1}^{p} c_k\right) v(t_n), \tag{3}$$

where the quantities $R(n)$, $u(n)$, and $v(t_n)$ are recursively computed as follows:

$$\begin{bmatrix} R(n) & u(n) \\ 0^T & v(t_n) \end{bmatrix} = T(n) \begin{bmatrix} \beta R(n-1) & \beta u(n-1) \\ X(t_n) & y(t_n) \end{bmatrix}. \tag{4}$$

$T(n)$ is a unitary matrix [1] of size $n \times n$ that performs a sequence of p Givens rotations. The quantity c_k in (3) represents the cosine term of the k^{th} rotation. For the initialization of the recursion in (4) the reader may refer to [4]. Equations (3) and (4) constitute *the QRD-RLS algorithm for the optimal residual computation.*

This algorithm uses the Givens rotation as a building block and it can be implemented by a fully parallel and pipelined triangular systolic array [7]. It has been proved that the QRD-based algorithms have good numerical properties [4]. Nevertheless, they are not very appropriate for VLSI implementation, because of the square root and the division operations that are involved in the Givens rotation and the forward-substitution required for the case of weight extraction (cf. (2)).

Several papers have proposed modifications in order to reduce the computational load involved in the original Givens rotation (for example [2],[3],[5]). These rotation-based algorithms are not rotations any more, since they do not exhibit the normalization property of the Givens rotation. Nevertheless, they can substitute the Givens rotation as the building block of the QRD algorithm, thus they can be treated as rotation algorithms in a wider sense:

Definition 1 *We will call* $\Re$otation algorithm *a Givens-rotation-based algorithm that can be used as the building block of the QRD algorithm.*

McWhirter has been able to compute the optimal residual of the RLS algorithm without square root operations [7] based on Gentleman's $\Re$otation [2]. A generalization of this result has appeared in [5]. Also, a fully-pipelined structure for weight extraction that circumvents the back-substitution divisions was derived independently in [8] and [9]. In both works the key idea is the use of a recursion for $R^{-H}(n)$ in (2):

$$\begin{bmatrix} R^{-H}(n) \\ \# \\ \# \end{bmatrix} = T(n) \begin{bmatrix} \frac{1}{\beta} R^{-H}(n-1) \\ \# \\ 0^T \end{bmatrix}. \tag{5}$$

No divisions are required for the evaluation of this recursion. Nevertheless, p divisions have to be evaluated during the initialization phase, where $R^{-H}(p)$

[1] A square matrix T is a unitary matrix if it has the property $T^H T = I$, where T^H denotes the Hermitian transpose matrix.

is computed from $R(p)$ with forward substitution. Equations (4), (2) and (5) constitute *the QRD-RLS algorithm for the optimal weight extraction.*
It has been shown that a square-root-free and division-free $\Re$otation does exist [3]. In this paper, we introduce a parametric family of square-root-free and division-free $\Re$otations. We will refer to this family of algorithms with the name *Parametric $\kappa\lambda$ $\Re$otation.* We will also say that a $\Re$otation algorithm is *a $\kappa\lambda$ $\Re$otation* if this algorithm is obtained by the *P*arametric $\kappa\lambda$ $\Re$otation with a choice of specific values for the parameters κ and λ. We employ the arguments in [5], [7] and [8] in order to design novel systolic architectures for the RLS algorithms that have less computational and circuitry complexity from the already known architectural implementations.
In Section 2, the *P*arametric $\kappa\lambda$ $\Re$otation is introduced. In Section 3, the RLS algorithms that are based on the *P*arametric $\kappa\lambda$ $\Re$otation are derived and the architectural implementations are considered for a specific $\kappa\lambda$ $\Re$otation. We conclude with Section 4.

2 SQUARE ROOT AND DIVISION FREE ALGORITHMS

In (4) the multiplication with the matrix $T(n)$ is equivalent to p consequent rotations. Each one of the rotations operates (for real valued data) as follows:

$$\begin{bmatrix} \alpha_1' & \alpha_2' & \cdots & \alpha_m' \\ 0 & \beta_2' & \cdots & \beta_m' \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \beta\alpha_1 & \beta\alpha_2 & \cdots & \beta\alpha_m \\ \beta_1 & \beta_2 & \cdots & \beta_m \end{bmatrix}, \qquad (6)$$

where

$$c = \frac{\beta\alpha_1}{\sqrt{\beta^2\alpha_1^2+\beta_1^2}}, \quad s = \frac{\beta_1}{\sqrt{\beta^2\alpha_1^2+\beta_1^2}} \qquad (7)$$

$$\alpha_1' = \sqrt{\beta^2\alpha_1^2+\beta_1^2} \qquad (8)$$

$$\alpha_j' = c\beta\alpha_j + s\beta_j, \quad \beta_j' = -s\beta\alpha_j + c\beta_j, \quad j = 2,3,\cdots,m \ . \qquad (9)$$

The Givens rotation is specified by the equations (7) to (9). We introduce the following data transformation:

$$\alpha_j = \frac{1}{\sqrt{l_a}}a_j, \quad \beta_j = \frac{1}{\sqrt{l_b}}b_j, \quad \alpha_j' = \frac{1}{\sqrt{l_a'}}a_j', \quad j = 1,2,\cdots,m \qquad (10)$$

$$\beta_j' = \frac{1}{\sqrt{l_b'}}b_j', \quad j = 2,3,\cdots,m. \qquad (11)$$

We seek the square root and division-free expressions for the transformed data $a_j', j = 1,2,\cdots,m,$ $b_j', j = 2,3,\cdots,m,$ l_a' and l_b' in terms of $a_j, b_j, j = 1,2,\cdots,m,$ l_a and l_b. By substituting (10)-(11) in (8)-(8) and solving for a_1', a_j' and b_j', we get

$$a_1' = \sqrt{\frac{l_a'}{l_a l_b}(l_b\beta^2 a_1^2 + l_a b_1^2)}. \qquad (12)$$

$$a'_j = \frac{l_b\beta^2 a_1 a_j + l_a b_1 b_j}{\sqrt{l_a l_b (l_b\beta^2 a_1^2 + l_a b_1^2)/l'_a}}, \quad b'_j = \frac{-b_1\beta a_j + \beta a_1 b_j}{\sqrt{(l_b\beta^2 a_1^2 + l_a b_1^2)/l'_b}} \quad j = 2,3,\cdots,m \ . \tag{13}$$

We will let l'_a and l'_b be equal to

$$l'_a = l_a l_b (l_b\beta^2 a_1^2 + l_a b_1^2)\kappa^2, \quad \text{and} \quad l'_b = (l_b\beta^2 a_1^2 + l_a b_1^2)\lambda^2, \tag{14}$$

where κ and λ are two parameters. By substituting (14) in (12)-(13) we obtain the expressions

$$a'_1 = \kappa(l_b\beta^2 a_1^2 + l_a b_1^2) \tag{15}$$

$$a'_j = \kappa(l_b\beta^2 a_1 a_j + l_a b_1 b_j), \quad b'_j = \lambda\beta(-b_1 a_j + a_1 b_j), \quad j = 2,3,\cdots,m. \tag{16}$$

If the evaluation of the parameters κ and λ does not involve any square root or division operations, the update equations (14)-(16) will be square root and division-free. In other words, every such choice of the parameters κ and λ specifies a square root and division-free $\Re$otation algorithm.

Definition 2 *Equations (14)-(16) specify* the Parametric $\kappa\lambda$ $\Re$otation algorithm. *Furthermore, a $\Re$otation algorithm will be called* a $\kappa\lambda$ $\Re$otation *if it is specified by (14)-(16) for specific square-root-free and division-free expressions of the parameters κ and λ.*

One can easily verify that the only one square root and division-free $\Re$otation in the literature to date [3] is a $\kappa\lambda$ $\Re$otation and is obtained for $\kappa = \lambda = 1$.

3 NOVEL RLS ALGORITHMS AND ARCHITECTURES

In this Section, we use the $\Re$otation described by the equations (14)-(16), in order to perform the QR decomposition of the data matrix $X(n)$. More specifically, we perform the triangularization in (4). We derive a parametric formula for the optimal residual $e_{RLS}(t_n)$ that involves no square root operations and only one division operation. We describe the systolic architectural implementation that is associated with the $\kappa\lambda$ $\Re$otation for which $\kappa = \lambda = 1$. Then, the systolic structure for the parallel computation of the weight vector $w(n)$ that was introduced in [8] and [9] is modified to provide a fast systolic implementation for weight vector extraction.

A Very Fast Algorithm for the RLS Optimal Residual Computation

Equation (4) symbolically can be written as

$$\begin{bmatrix} \sqrt{\frac{1}{l_1}} & & & & \\ & \sqrt{\frac{1}{l_2}} & & & \\ & & \ddots & & \\ & & & \sqrt{\frac{1}{l_p}} & \\ & & & & \sqrt{\frac{1}{l_q}} \end{bmatrix} \begin{bmatrix} \beta a_{11} & \beta a_{12} & \cdots & \beta a_{1p} & \beta a_{1,p+1} \\ & \beta a_{22} & \cdots & \beta a_{2p} & \beta a_{2,p+1} \\ & & \ddots & \vdots & \vdots \\ & & & \beta a_{pp} & \beta a_{p,p+1} \\ b_1 & b_2 & \cdots & b_p & b_{p+1} \end{bmatrix}$$

$$\xrightarrow{T(n)} \begin{bmatrix} \sqrt{\frac{1}{l'_1}} & & & & \\ & \sqrt{\frac{1}{l'_2}} & & & \\ & & \ddots & & \\ & & & \sqrt{\frac{1}{l'_p}} & \\ & & & & \sqrt{\frac{1}{l'_q}} \end{bmatrix} \begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1p} & a'_{1,p+1} \\ & a'_{22} & \cdots & a'_{2p} & a'_{2,p+1} \\ & & \ddots & \vdots & \vdots \\ & & & a'_{pp} & a'_{p,p+1} \\ 0 & 0 & \cdots & 0 & b^{(p)}_{p+1} \end{bmatrix}. \tag{17}$$

Equations (14)-(16) imply that the first $\Re$otation is specified as follows:

$$l'_1 = l_1 l_q (l_q \beta^2 a_{11}^2 + l_1 b_1^2)\kappa_1^2 \tag{18}$$

$$l_q^{(1)} = (l_q \beta^2 a_{11}^2 + l_1 b_1^2)\lambda_1^2, \tag{19}$$

$$a'_{1j} = \kappa_1 (l_q \beta^2 a_{11} a_{1j} + l_1 b_1 b_j), \quad j = 1, 2, \cdots, p+1 \tag{20}$$

$$b_j^{(1)} = \lambda_1 \beta(-b_1 a_{1j} + a_{11} b_j), \quad j = 2, 3, \cdots, p+1. \tag{21}$$

Similarly, the i^{th} $\Re$otation is specified as follows:

$$l'_i = l_i l_q^{(i-1)} (l_q^{(i-1)} \beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2})\kappa_i^2 \tag{22}$$

$$l_q^{(i)} = (l_q^{(i-1)} \beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2})\lambda_i^2, \tag{23}$$

$$a'_{ij} = \kappa_i (l_q^{(i-1)} \beta^2 a_{ii} a_{ij} + l_i b_i^{(i-1)} b_j^{(i-1)}), \quad j = i, i+1, \cdots, p+1 \tag{24}$$

$$b_j^{(i)} = \lambda_i \beta(-b_i^{(i-1)} a_{ij} + a_{ii} b_j^{(i-1)}), \quad j = i+1, i+2, \cdots, p+1, \tag{25}$$

where $i = 2, 3, \cdots, p$ and $b_j^{(0)} = b_j$, $j = 1, \cdots, p+1$. In [1] the following lemma is proved:

Lemma 1 *If the Parametric $\kappa\lambda$ $\Re$otation is used in the QRD-RLS algorithm, the optimal residual is given by the expression*

$$e_{RLS}(t_n) = -\left(\prod_{j=1}^{p-1} \lambda_j \beta a_{jj}\right) \frac{\kappa_p \beta a_{pp}}{\lambda_p a'_{pp}} b_{p+1}^{(p)} v, \tag{26}$$

where $v = \frac{1}{sqrtl_q}$ if p is an even number and $v = \sqrt{l_q}$ if p is an odd number.
□

Here, l_q is a free variable. If we choose $l_q = 1$ we get $v = 1$ for both even and odd values of p and we can avoid the square root operation. We can see that for a recursive computation of (26) only one division operation is needed at the last step of the recursion. This compares very favorably with the *fast algorithm* (based on a square root free $\Re$otation) that requires one division for every recursion step, as well as with the original approach (3), which involves one division and one square root operation for every recursion step.

For this reason, the algorithms and architectures based on the *P*arametric $\kappa\lambda$ $\Re$otation will be called *very fast algorithms* and *architectures*.
The division operation in (26) cannot be avoided by proper choice of expressions for the parameters κ and λ. This is restated by the following lemma, which is proved in [1]:

Lemma 2 *If a $\kappa\lambda$ $\Re$otation is used, the RLS optimal residual evaluation will involve the evaluation of at least one division.* □

A Systolic Architecture for the Optimal RLS Residual Evaluation

McWhirter has used a systolic architecture for the implementation of the QR decomposition [7]. This architecture is modified, so that equations (18)-(26) be evaluated for the special case of

$$\kappa_i = \lambda_i = 1, i = 1, 2, \cdots, p \text{ and } l_q = 1.$$

The systolic array, as well as the memory and the communication links of its components, are depicted in figure 1 [2]. The boundary cells (cell number 1) are responsible for evaluating (22) and (23), as well as the coefficients $\bar{c}_i = l_q^{(i-1)} a_{ii}$ and $\bar{s}_i = l_i b_i^{(i-1)}$ and the partial products $e_i = \prod_{j=1}^{i} (\beta a_{jj})$. The internal cells (cell number 2) are responsible for evaluating (24) and (25). Finally, the output cell (cell number 3) evaluates (26). The functionality of each one of the cells is described in figure 1. We will call this systolic array $S1.1$.
On table 1, we collect some features of the systolic structure $S1.1$ and the two structures, $S1.2$ and $S1.3$, in [7] that are pertinent to the circuitry complexity. Note that $S1.2$ implements the square-root-free QR decomposition algorithm introduced by McWhirter [7], while $S1.3$ is the systolic implementation of the QR decomposition based on the original Givens rotation. On table 1, the complexity per processor cell and the number of required processor cells are indicated for each one of the three different kinds of processors [3]. One can easily observe that $S1.1$ requires the implementation of only one division operator and no square root operator, $S1.2$ requires the implementation of p division operators and no square root operator, while $S1.3$ requires the implementation of p division and p square root operators. This reduction of the complexity in terms of division and square root operators is penalized with the increase of the number of the multiplications and the communication links that are required.
Apart from the circuit complexity that is involved in the implementation of the systolic structures, another feature of the computational complexity is the number of *operations-per-cycle*. This number determines the minimum

[2] Note the aliases: $l_q^{(i-1)} \equiv \sigma_{in}, l_q^{(i)} \equiv \sigma_{out}, l_i \equiv l, a_{ij} \equiv r, b_j^{(i-1)} \equiv b_{in}, b_j^{(i)} \equiv b_{out}$, $e_{i-1} \equiv e_{in}, e_i \equiv e_{out}$.
[3] The multiplications with the constants β and β^2 are not encountered.

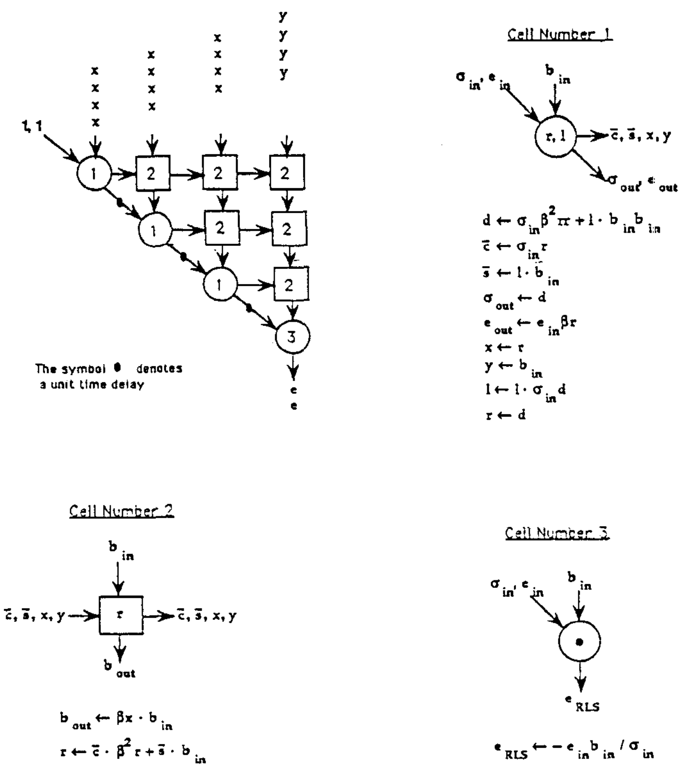

Figure 1: $S1.1$: This systolic array computes the RLS optimal residual. It implements the algorithm that is based on the $\kappa\lambda$ $\Re$otation for which $\kappa = \lambda = 1$.

	$S1.1$			$S1.2$			$S1.3$		
cell	1	2	3	1	2	3	1	2	3
number of	p	$\frac{p(p+1)}{2}$	1	p	$\frac{p(p+1)}{2}$	1	p	$\frac{p(p+1)}{2}$	1
sq.rt	-	-	-	-	-	-	1	-	-
div.	-	-	1	1	-	-	1	-	-
mult.	9	4	1	5	3	1	4	4	1
i/o	9	10	4	6	8	3	5	6	3
operations per cycle	max{1 *div.* + 1 *mult.*, 9 *mult.*}			1 *div.* + 5 *mult.*			1 *sq.rt.* + 1 *div.* +4 *mult.*		

Table 1: RLS residual computational complexity.

required delay between two consequent sets of input data. For the structures $S1.2$ and $S1.3$ the boundary cell (cell number 1) constitutes the bottleneck of the computation and therefore it determines the operations-per-cycle that are shown on table 2. For the structure $S1.1$ either the boundary cell or the output cell are the bottleneck of the computation. This is reflected on the corresponding expression on table 1.

Algorithm and a Systolic Architecture for the Optimal RLS Weight Extraction

We have seen in Section 1 that we can calculate the optimal weight vector in a recursive way: We can compute $R^{-H}(n)$ by using the same rotation (5) that computes $R(n)$ and then use parallel multiplication (2) for computing $w^T(n)$. This algorithm can be implemented by a systolic structure, which is a variation of the one used for the residual calculation and it was derived independently by Shepherd et al. [8] and Tang et al. [9]. It is a fully pipelined array that can operate in two distinct modes, 0 and 1. The initialization phase consists of $2p$ steps for each processor. During the first p steps the processors operate in mode 0 in order to calculate a full rank matrix R. During the following p steps, the processors operate in mode 1 in order to compute R^{-H}, by performing a task equivalent to forward substitution. After the initialization phase the processors operate in mode 0. In [8] one can find the systolic array implementations that are based both on the original Givens rotation and the Gentleman's variation of the square-root-free $\Re$otation. We will call these two structures $S2.3$ and $S2.2$ respectively.
In figure 2, we present the systolic structure $S2.1$, which is based on the $\kappa\lambda$ $\Re$otation with $\kappa = \lambda = 1$. The functionality of the processing cells, as well as their communication links and their memory contents, are given in figure 2. On table 2 we collect some computational complexity metrics for the systolic arrays $S2.1$, $S2.2$ and $S2.3$, when they operate in mode 0[4]. The conclusions we can draw are similar to the ones we had for the circuits that calculate the optimal residual. We should also note that $S2.1$ does require the implementation of division operators in the boundary cells, since these operators are used during the initialization phase. Nevertheless, after the initialization phase the circuit will not suffer from any time delay caused by division operations.
The computational botlleneck of all three structures, $S2.1$, $S2.2$ and $S2.3$, is the boundary cell, thus it determines the operations-per-cycle metric.

4 CONCLUSIONS AND FURTHER RESEARCH

We introduced *the Parametric* $\kappa\lambda$ *$\Re$otation*, which is a square-root-free and division-free algorithm. We reviewed the systolic architectures that implement QRD-RLS algorithms and they are based on the Givens rotation,

[4]The multiplications with the constants $\beta, \beta^2, 1/\beta$ and $1/\beta^2$, as well as the communication links that drive the mode bit, are not encountered.

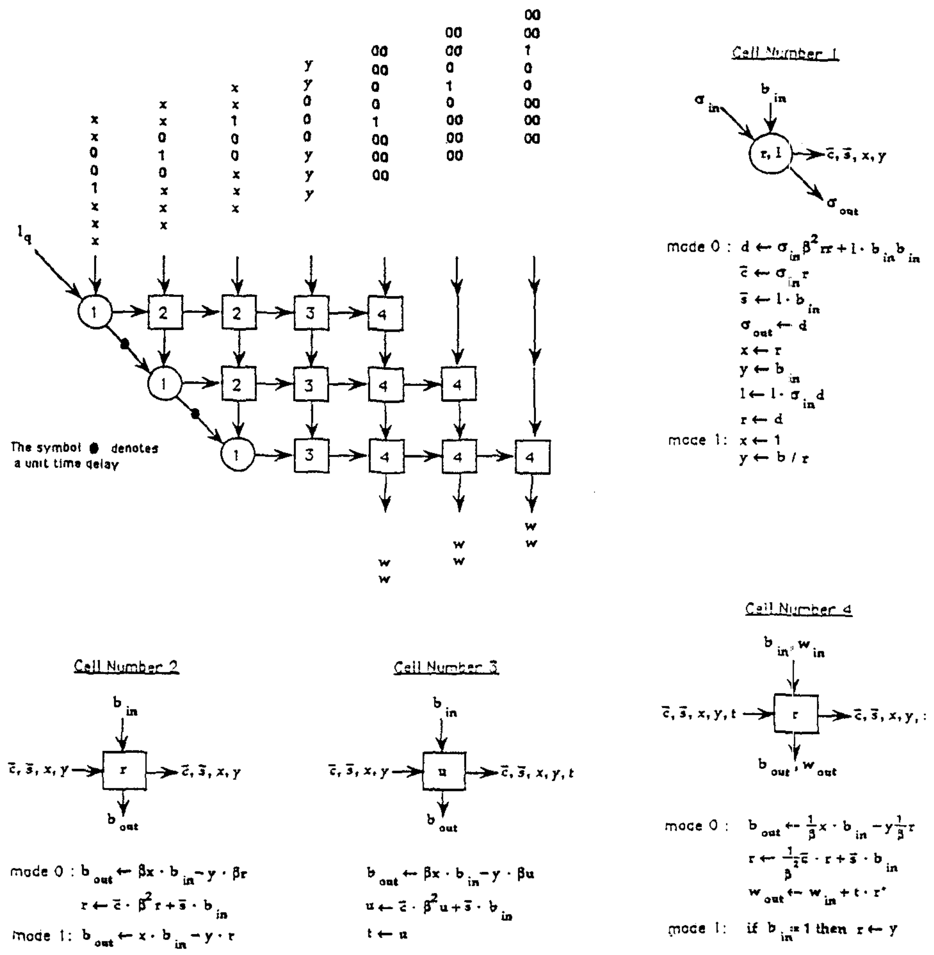
The symbol ● denotes a unit time delay
Cell Number 1
mode 0 : d ← σ_in β² r r + l · b_in b_in
c̄ ← σ_in r
s̄ ← l · b_in
σ_out ← d
x ← r
y ← b_in
l ← l · σ_in d
r ← d
mode 1: x ← 1
y ← b / r
Cell Number 2
mode 0 : b_out ← βx · b_in − y · βr
r ← c̄ · β² r + s̄ · b_in
mode 1: b_out ← x · b_in − y · r
Cell Number 3
b_out ← βx · b_in − y · βu
u ← c̄ · β² u + s̄ · b_in
t ← u
Cell Number 4
mode 0 : b_out ← (1/β) x · b_in − y (1/β) r
r ← (1/β²) c̄ · r + s̄ · b_in
w_out ← w_in + t · r*
mode 1: if b_in = 1 then r ← y

Figure 2: $S2.1$: This systolic array computes the RLS optimal weight vector. It implements the algorithm that is based on the $\kappa\lambda$ $\Re$otation for which $\kappa = \lambda = 1$.

	$S2.1$				$S2.2$				$S2.3$			
cell	1	2	3	4	1	2	3	4	1	2	3	4
number of	p	$\frac{(p-1)p}{2}$	p	$\frac{p(p+1)}{2}$	p	$\frac{(p-1)p}{2}$	p	$\frac{p(p+1)}{2}$	p	$\frac{(p-1)p}{2}$	p	$\frac{p(p+1)}{2}$
sq.rt	-	-	-	-	-	-	-	-	1	-	-	-
div.	-	-	-	-	1	-	-	-	1	-	-	-
mult.	8	4	4	5	5	3	3	4	4	4	4	5
i/o	7	10	11	14	6	8	9	12	3	6	7	10
operations per cycle	*8 mult.*				*1 div. + 5 mult.*				*1 sq.rt. + 1 div. +4 mult.*			

Table 2: RLS weight extraction computational complexity (mode 0).

as well as the Gentleman'n square root free $\Re$otation. We introduced novel architectures that are based on the $\kappa\lambda$ $\Re$otation for $\kappa = \lambda = 1$ and we made a comparative study of all three kinds of architectures. We observed that considerable improvements can be obtained for the implementation of the QRD-based RLS algorithms.
We have not stated any arguments concerning the optimality for the choice of the values of the parameters κ, λ. We underline the fact that both the $\kappa\lambda$ $\Re$otation algorithms expose computational complexity advantages, compared to the original Givens rotation, with the cost of the denormalization of the latter. Consequently, the architectures based on them should incorporate some additional logic that ensures numerical stability. Actually, Gotze and Schwiegelshohn have shown that the $\kappa\lambda$ $\Re$otation with $\kappa = \lambda = 1$ can be stabilized with negligible overhead [3]. Finally, a study on the dynamic range needs to be made before we be able to infer any optimality statement about the implementation cost.

References

[1] E. Frantzeskakis and K. J. R. Liu, "A Class of Square Root and Division Free Algorithms and Architectures for QRD-Based Adaptive Signal Processing," *Submitted to IEEE Trans. on Signal Processing,* 1992.

[2] W. M. Gentleman, "Least Squares Computations by Givens Transformations without Square Roots," *J. Inst. Maths. Applics.*, vol. 12, pp. 329–336, 1973.

[3] J. Gotze and U. Schwiegelshohn, "A Square Root and Division Free Givens Rotation for Solving Least Squares Problems on Systolic Arrays," *SIAM J. Scie. and Stat. Comput.*, vol. 12, no. 4, pp. 800–807, July 1991.

[4] S. Haykin, *Adaptive Filter Theory,* 2nd ed., Prentice Hall, 1991.

[5] S. F. Hsieh, K. J. R. Liu and K. Yao, A Unified Approach for QRD-Based Recursive Least-Squares Estimation without Square Routs", *To appear in IEEE Trans. on Signal Processing,* May 1993. Also in *Proc. IEEE/ICASSP*, Toronto, May 1991, pp. 1017–1020.

[6] J. G. McWhirter, "Recursive Least-Squares Minimization using a Systolic Array," *Proc. of SPIE, Real Time Signal Processing VI*, 1983, vol. 431, pp. 105–112.

[7] T. J. Shepherd, J. G. McWhirter and J. E. Hudson, "Parallel Weight Extraction from a Systolic Adaptive Beamforming," *Mathematics in Signal Processing II*, 1990.

[8] C. F. T. Tang and K. J. R. Liu, "A VLSI Algorithm and Architecture for CRLS Adaptive Beamforming," *Proc. of the 25th International Conference on Information Sciences and Systems*, Baltimore, March 1991, pp. 862–867.